UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/563,313	12/29/2005	Sergey N. Zheltov	42P21486	6943

45209 7590 05/11/2009

INTEL/BSTZ
BLAKELY SOKOLOFF TAYLOR & ZAFMAN LLP
1279 OAKMEAD PARKWAY
SUNNYVALE, CA 94085-4040

EXAMINER	
KHATRI, ANIL	

ART UNIT	PAPER NUMBER
2191	

MAIL DATE	DELIVERY MODE
05/11/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/563,313	ZHELTOV ET AL.	
	Examiner	Art Unit	
	Anil Khatri	2191	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *29 December 2005*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-30* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-30* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *29 December 2005* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *4/3/06*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

Specification

The title of the invention is not descriptive. A new title is required that is clearly indicative of the invention to which the claims are directed.

The following title is suggested: "A Platform Independent Binary Instrumentation Method and Memory Allocation".

Claim Rejections - 35 USC § 101

35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

Analysis: Claims 1-30 disclosed by the applicant as being a "system and method for binary instrumentation…". Since the claims are each a series of steps to be performed on a computer the processes must be analyzed to determine whether they are statutory under 35 USC 101.

Examiner interprets that claims 1-30 are non-statutory because claim recites computer program which is program per se i.e. the description or expressions of the program are not physical things nor are they statutory process as they do not act being performed. Computer programs do not define any structural and functional interrelationship between the computer program and other claimed aspect of the invention which permits the computer program's functionality could be realized. Therefore, computer program is merely a set of instructions capable of being executed

by a computer, the computer program itself is not a process. Therefore, claims 1-30 are merely allocating memory, managing region and data structure which is not able to produce a useful results and practical application and are non-statutory and rejected under 35 USC 101.

Analysis: Claims 11-20 disclosed by the applicant as being a "a machine accessible medium…". Since the claims are each a series of steps to be performed on a computer the processes must be analyzed to determine whether they are statutory under 35 USC 101.

Examiner interprets that claims 11-20 are not limited to tangible embodiments in view of applicant's disclosure, specification pages 5-6 medium is not limited to tangible embodiments, instead being defined as including both tangible embodiments (e.g., [computer readable medium]) and intangible embodiments (e.g., [transmission media, radio frequency (RF), infrared (IR), a carrier wave, telephone line, a signal, etc.]). As such, the claim is not limited to statutory subject matter and is therefore non-statutory. To overcome this type of 101 rejection the claims need to be amended to include only the physical computer media and not a transmission media or other intangible or non-functional media. For the specification, carrier medium and transmission media would be not statutory but storage media would be statutory.

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

Claims 1-30 are rejected under 35 U.S.C. 103(a) as being unpatentable over Stimac et al USPN 4,926,322 in view of Dmitriev USPN 7,293,260

Regarding claims 1, 11 and 21

Stimac et al teaches

allocating a contiguous memory region (columns 5-6, lines 55-57 and lines 1-3, the EMM memory manager program is activated when the computer is first started up, and it lays the groundwork for the expanded memory's operation. A key part of its task is to find an unused area in the computer's memory space into which it can map the bank-switch memory. It requires a full 64K work area, called a page frame. As can be seen from the general memory allocation shown in FIG. 1, the D and E blocks of memory are obviously good candidates for the location of the page frame; however, the EMM can place the page frame in the C block as well. The exact location doesn't matter, as long as it doesn't interfere with any other use of the memory address space. Also, the 64K page frame doesn't have to be placed on a memory block boundary. For example, it can begin at the segment address C400 and extend up through the rest of the C block and into the first 16K of the D block);

filling the memory region with at least one copy of an interceptor function (column 19, lines 13-22, the System ROM Int 15h Move Block function (AH=87h) provides access to extended memory for real mode MS-DOS programs. Because the ROM's code for implementing the function executes in protected mode, CEMM intercepts Move Block function calls (int 15h with AH=87H) and implements the Move Block function in the VDM code. The VDM Move Block function uses a DWORD REP MOVS instruction for faster execution. For more information on the System ROM Move Block function, see the COMPAQ Deskpro 386 Technical Reference);

initializing a first data structure with at least a starting address, length of the allocated memory region, and a reference to a second data structure (column 3,lines 47-58, the entire 1 megabyte address space of the 8086 can be thought of as divided into 16 blocks of 64K each. Each of these blocks of memory can be identified by the high-order (most significant) hex digit that all addresses in that block share. Thus, the first 64K of memory can be called the 0 block, since all addresses in that block are 0xxxx (in five-digit absolute address notation) or 0xxx:xxxx (in segmented address notation). Likewise, the second block is the 1-block, since all addresses in that 64K begin with 1. In the 1 megabyte address space, there are 16 blocks of 64K, which may be designated the 0-block through the F-block (in hexadecimal notation). Stimac et al does not teach explicitly instrumentation, however Dmitriev teaches storing an address of an original function in a current element of the second data structure upon request for instrumentation (columns 2-3, lines 65-67 and lines 1-9, one embodiment of the

present invention provides a system that facilitates configuring selected methods for instrument-based profiling at run-time. The system first identifies a root method in a target application, wherein only methods that are reachable from the root method during execution of the target application are to be instrumented. The system then instruments the root method. Next, while subsequently executing a given instrumented method, the system determines if the given instrumented method is about to be executed for the first time. If so, the system instruments any methods that are called by the given instrumented method, are loaded, and have not been instrumented before). Dmitriev further teaches providing a starting address of a copy of the interceptor function upon request for instrumentation (column 10, lines 8-24, technically, runtime hotswapping of a method is not such a challenging issue as it might seem. The task is simplified by the fact that our current policy for switching between the original and the instrumented method version is "all active invocations of the original method complete as is, and all calls to this method that happen after hotswapping go to the instrumented method version." In other words, if a method that we want to instrument is currently running, we don't attempt to switch execution to the instrumented code immediately. The latter feature does not currently seem to be very valuable in practice, since methods that never exit (for example because they spin in an endless cycle) are rare and can be addressed separately if needed. Also, if immediate switching is performed, the instrumented code would generate a number of "method exit" events that are not matched with corresponding "method entry"

ones).Therefore, it would have been obvious to a person of ordinary skill in the art at the time of the invention was made to incorporate instrumentation in allocating memory. The modification would have been obvious because one of ordinary skill in the art would have been motivated to combine teaching into allocating memory for optimization and save overheads during execution.

Regarding claims 2, 12 and 22

Stimac et al teaches

allocating a memory region, filling the memory region, and initializing the fast data structure are performed upon an initial request for instrumentation (columns 5-6, lines 55-57 and lines 1-3, the EMM memory manager program is activated when the computer is first started up, and it lays the groundwork for the expanded memory's operation. A key part of its task is to find an unused area in the computer's memory space into which it can map the bank-switch memory. It requires a full 64K work area, called a page frame. As can be seen from the general memory allocation shown in FIG. 1, the D and E blocks of memory are obviously good candidates for the location of the page frame; however, the EMM can place the page frame in the C block as well. The exact location doesn't matter, as long as it doesn't interfere with any other use of the memory address space. Also, the 64K page frame doesn't have to be placed on a memory block boundary. For example, it can begin at the segment address C400 and extend up through the rest of the C block and into the first 16K of the D block).

Regarding claims 3, 13, and 23

Stimac et al teaches

allocating a memory region, filling the memory region, and initializing the first data structure are performed if all interceptor function copies of currently allocated memory regions are associated with previous requests for instrumentation (columns 5-6, lines 55-57 and lines 1-3, the EMM memory manager program is activated when the computer is first started up, and it lays the groundwork for the expanded memory's operation. A key part of its task is to find an unused area in the computer's memory space into which it can map the bank-switch memory. It requires a full 64K work area, called a page frame. As can be seen from the general memory allocation shown in FIG. 1, the D and E blocks of memory are obviously good candidates for the location of the page frame; however, the EMM can place the page frame in the C block as well. The exact location doesn't matter, as long as it doesn't interfere with any other use of the memory address space. Also, the 64K page frame doesn't have to be placed on a memory block boundary. For example, it can begin at the segment address C400 and extend up through the rest of the C block and into the first 16K of the D block) and (column 19, lines 13-22, the System ROM Int 15h Move Block function (AH=87h) provides access to extended memory for real mode MS-DOS programs. Because the ROM's code for implementing the function executes in protected mode, CEMM intercepts Move Block function calls (int 15h with AH=87H) and implements the Move Block function in the VDM code. The VDM Move Block function uses a DWORD REP

MOVS instruction for faster execution. For more information on the System ROM Move Block function, see the COMPAQ Deskpro 386 Technical Reference).

Regarding claims 4, 14 and 24

Dmitriev teaches

duplicating the first data structure to associate each new copy of the first data structure with each newly allocated memory region (column 12, lines 1-19, a new internal method object is created for each method scheduled for instrumentation, and then the VM internal bytecode rewriting code is used to create "holes" in the bytecode copy, possibly re-computing jump operations that cross these holes. Then we fill in the holes with calls to our methodEntry( ), methodExit( ), etc. methods (see the next section). Since calls in Java bytecodes go via constant pool references, we have to check the constant pool for existing references to the injected instrumentation methods. If there are none, we create a new, extended constant pool object with added method references, and attach it to the instrumented method version to prevent it from being garbage collected. The instrumented method copy is, in turn, attached to its class object, using a special instrumented methods array that we introduced, which is similar to the standard methods array holding references from the class to its methods. This is illustrated in FIG. 2, where it is shown what happens if one of two class's methods, m1( ), is instrumented).

Regarding claims 5, 15 and 25

Dmitriev teaches

second data structure comprises elements to store addresses of original functions instrumentation was requested for (column 23, lines 30-43, in one embodiment of the present invention, all bytecode scanning and other code analysis operations are performed at the agent side. The code at the target VM (server) side sends to the agent messages such as "class loaded", "initial list of classes loaded by the VM", etc. and receives messages containing lists of methods to instrument. It is worth noting that once any code has been instrumented and thus CPU profiling started, further code analysis and instrumentation upon class loading, which may be relatively time-consuming, can affect profiling results quite significantly. To prevent this, every time a class load event occurs, our server-side code records a special "thread suspend" profiling event. The "thread resume" event is recorded just before returning from the class load hook to the TA code. This compensates for the time spent in code analysis and instrumentation).

Regarding claims 6, 16 and 26

Dmitriev teaches

maintaining the current element of the second data structure to establish a correspondence between the original function and a provided address of an interceptor function copy (column 24, lines 31-41, the technique itself works as follows. 1. The

procedure starts when the root method's class is loaded (we can intercept this event using a class load hook). Obtain the list of all classes currently loaded by the VM and create mirror data structures in the same way as in Scheme A. Also create an expandable global boolean array method Invoked [ ] that indicates whether any instrumented method has been invoked at least once. 2. Instrument the root method m( ), and mark it as instrumented in the C's mirror.

Regarding claims 7, 17 and 27

Dmitriev teaches

selecting a next successive element of the second data structure as the current element for each new request for instrumentation (column 12, lines 42-46, after the second operation is complete, the application threads are resumed, and the next call the target application makes to an instrumented method, goes to its new version. The invocations of original method versions that are currently active complete as is…).

Regarding claims 8, 18 and 28

Stimac et al teaches

a reference to the second data structure comprises at least one of a memory address of and an index to the second data structure (figure 11, column 8, lines 38-45, a selector is an index into a segment descriptor table; that is, it is a segment number. Each entry in a segment descriptor table contains the base address of a segment. The processor adds the offset to the segment's base address to produce a 32-bit linear

address. If paging is not enabled, the processor considers the linear address to be the physical address and emits it on the address pins).

Regarding claims 9, 19 and 29

Dmitriev teaches

the starting address of a copy of the interceptor function is provided in a direct correspondence with the current element of the second data structure (column 12, lines 1-19, a new internal method object is created for each method scheduled for instrumentation, and then the VM internal bytecode rewriting code is used to create "holes" in the bytecode copy, possibly re-computing jump operations that cross these holes. Then we fill in the holes with calls to our methodEntry( ), methodExit( ), etc. methods (see the next section). Since calls in Java bytecodes go via constant pool references, we have to check the constant pool for existing references to the injected instrumentation methods. If there are none, we create a new, extended constant pool object with added method references, and attach it to the instrumented method version to prevent it from being garbage collected. The instrumented method copy is, in turn, attached to its class object, using a special instrumented methods array that we introduced, which is similar to the standard methods array holding references from the class to its methods. This is illustrated in FIG. 2, where it is shown what happens if one of two class's methods, m1( ), is instrumented).

Regarding claims 10, 20 and 30

Stimac et al teaches

the interceptor function comprises obtaining an address being currently executed (column 19, lines 13-22, the System ROM Int 15h Move Block function (AH=87h) provides access to extended memory for real mode MS-DOS programs. Because the ROM's code for implementing the function executes in protected mode, CEMM intercepts Move Block function calls (int 15h with AH=87H) and implements the Move Block function in the VDM code. The VDM Move Block function uses a DWORD REP MOVS instruction for faster execution. For more information on the System ROM Move Block function, see the COMPAQ Deskpro 386 Technical Reference);

retrieving, from a corresponding copy of the first data structure, the starting address of a memory region that contains the address being currently executed (columns 4-5, lines 57-67 and 1-2, however, it is still possible for a program to make some use of extended memory. One way to do this is for a program to use some of the services provided by the computer's built-in ROM-BIOS programs. One of these services transfers blocks of data, in whatever size is needed, between the extended memory and conventional memory. An example of a program that uses the BIOS's transfer service to use extended memory is the virtual disk utility called VDISK which has been a part of DOS since version 3.0. When VDISK is activated with the extended memory, it uses the BIOS transfer service to move data into and out of extended

memory without VDISK needing to work in protected mode or directly manipulate the extended memory area);

fetching the reference to the second data structure from the copy of the first data structure (column 10, lines 6-18, the descriptors in a task's Local Descriptor Table (LDT) and Global Descriptor Table (GDT) define the task's logical address space. The segments defined in these tables are theoretically addressable, because the descriptor tables provide the information necessary to compute a segment's address. However, an addressable segment may not be accessible to a particular operation because of the additional protection checks made by the 80386. The 80386 checks every segment reference (whether generated by the execution of an instruction or an instruction fetch) to verify that the reference is consistent with the protection attributes of the segment as described below);

computing an index to the second data structure as the fetched reference to the second data structure added to the difference between the address being currently executed and the retrieved starting address, the difference divided by the size of the interceptor function (column 16, lines 8-20, the VDM interrupt handler must be able to differentiate between these two conditions. The 386 VDM does so by taking advantage of the fact that the CPU puts an error code onto the stack when generating a Page Fault. Since the top of the ring 0 stack is a known value (it is taken from the TSS when the transition from virtual mode to protected mode occurs), checking for the

presence of the error code is simply a matter of measuring the depth of the stack. The stack is one word (the error code) deeper when the interrupt has been entered via processor exception than when it has been entered via hardware interrupt. Other processor exception/hardware interrupt conflicts may be handled in the same manner);and

reading, from the second data structure indexed with the computed index, the address of an original function to pass control to (see figures 3-5, column 7, lines 12-21, FIG. 3 shows in functional terms how the 80386 microprocessor translates a linear address to a physical address when paging is enabled. The processor uses the upper 10 bits of the linear address as an index into the directory. The selected directory entry contains the address of a page table. The processor adds the middle 10 bits of the linear address to the page table address to index the page table entry that describes the target page. Adding the lower 12 bits of the linear address to the page address produces the 32-bit physical address).

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Anil Khatri whose telephone number is 571-272-3725. The examiner can normally be reached on M-F 8:30-5:00 PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Wei Zhen can be reached on 571-272-3708. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

***

/Anil Khatri/

Primary Examiner, Art Unit 2191